

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class AckReply extends Reply {

    public AckReply(Command c) {
        this.seqNo = c.seqNo;
        this.len = 0;
    }

    public AckReply(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public int getMessageType(){
        return ACK_REPLY;
    }

    public byte[] cerealize() throws IOException {
        ByteArrayOutputStream bos= new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(bos);
        this.cerealize(dos);
        return bos.toByteArray();
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
    }

    public String toString(){
        return "<%=msgType=AckReply%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + ">";
    }
}
```

02/24/02
16:03:26

nmc\common\Command.java

1

```
package nmc.common;

public abstract class Command extends Message {
    private static int nextSeqNo = 1;
    public static synchronized int getNextCmdSeqNo() {
        return nextSeqNo++;
    }
}
```

```
package nmc.common;

import java.io.IOException;
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;

public class DecerealizingFactory extends Object {

    public static Message parse(byte[] buf) throws IOException {
        Message retval = null;
        System.out.println("Hey, we're in Decerealizing Factory!");
        System.out.println("buf.length=" + buf.length);
        ByteArrayInputStream bis= new ByteArrayInputStream(buf);
        DataInputStream dis = new DataInputStream(bis);
        int msgType = dis.readInt();
        switch (msgType) {
            case Message.SET_CURRENT_LIMIT_CMD:
                retval = (Message)new SetCurrentLimitCmd(dis);
                break;
            case Message.ACK_REPLY:
                retval = (Message)new AckReply(dis);
                break;
            case Message.POLL_CMD:
                retval = (Message)new PollCmd(dis);
                break;
            case Message.POLLED_DATA_REPLY:
                System.out.println("DecerealizingFactory reconstructing "
                    + "PolledDataReply...");
                if (dis==null)
                    System.out.println("dis==null");
                retval = (Message)new PolledDataReply(dis);
                break;
            case Message.INITIALIZE_INSTRUMENT_CMD:
                System.out.println("DecerealizingFactory reconstructing "
                    + "InitializeInstrumentCmd...");
                if (dis==null)
                    System.out.println("dis==null");
                retval = (Message)new InitializeInstrumentCmd(dis);
                break;
            case Message.INITIALIZE_INSTRUMENT_REPLY:
                System.out.println("DecerealizingFactory reconstructing "
                    + "InitializeInstrumentReply...");
                if (dis==null)
                    System.out.println("dis==null");
                retval = (Message)new InitializeInstrumentReply(dis);
                break;
            default:
                System.out.println("bad msgType (" + msgType +
                    ") encountered when constructing message");
                break;
        }
        return retval;
    }
}
```

```
package nmc.common;

import nmc.platform.FifoQueue;
import java.io.IOException;

import nmc.platform.DatagramIoPort;
import nmc.common.*;

public class InboundMessageQ implements Runnable {
    private FifoQueue q = new FifoQueue(50);
    private DatagramIoPort dgmRx;

    Thread myThread;

    public InboundMessageQ(int myRxPortNumber) throws IOException {
        dgmRx = DatagramIoPort.GetRcvPort(myRxPortNumber);
        // we won't get here if an IOException is thrown
        myThread = new Thread(this);
    }

    public Message blockingGet() {
        return (Message)q.blockingDequeue();
    }

    public void getOne() throws IOException {
        System.out.println("Inbound blocking on socket read...");
        byte[] rcvdData = dgmRx.blockingRx();
        Message m = DecerealizingFactory.parse(rcvdData);
        q.enqueue(m);
        System.out.println("received: " + m.toString());
    }

    public synchronized void startInboundMessaging() {
        myThread = new Thread(this);
        myThread.start();
    }

    public void run() {
        System.out.println("InboundMessage thread starting...");
        try {
            while (true) {
                getOne();
            }
        } catch (IOException e) {
            System.out.println("Inbound caught exception, stopping. e="
                               + e);
        }
    }

    public boolean isEmpty() {
        return q.isEmpty();
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class InitializeInstrumentCmd extends Command {

    public int channelNumber;

    public InitializeInstrumentCmd(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public InitializeInstrumentCmd(int channelNumber) {
        this.seqNo = getNextCmdSeqNo();
        this.len = 8;
        this.channelNumber = channelNumber;
    }

    public int getMessageType() {
        return INITIALIZE_INSTRUMENT_CMD;
    }

    public byte[] cerealize() throws IOException {
        ByteArrayOutputStream bos= new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(bos);
        this.cerealize(dos);
        return bos.toByteArray();
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
        s.writeInt(channelNumber);
    }

    public void decerealize(DataInputStream s) throws IOException {
        super.decerealize(s);
        channelNumber = s.readInt();
    }

    public String toString(){
        return "<%msgType=InitializeInstrumentCmd%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%channelNumber="
            + this.channelNumber
            + ">%";
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class InitializeInstrumentReply extends Reply {
    int channelNumber;
    boolean endOfData;
    int subseqNo;
    int seconds;

    public InitializeInstrumentReply(Message m, int seconds) {
        InitializeInstrumentCmd iic = (InitializeInstrumentCmd)m;
        this.seqNo = iic.seqNo;
        this.len = 4+1+4+4;
        this.channelNumber = iic.channelNumber;
        this.endOfData = true;
        this.subseqNo = 0;
        this.seconds = seconds;
    }

    public InitializeInstrumentReply(DataInputStream dis) throws IOException {
        System.out.println("in ctor(dis)");
        //      this.decerealize(dis);
        //      decerealize(dis);
        //      }

        //      public void decerealize(DataInputStream dis) throws IOException {
        System.out.println("about to superdecereal");
        super.decerealize(dis);
        System.out.println("next");
        channelNumber = dis.readInt();
        System.out.println("next");
        endOfData = dis.readBoolean();
        System.out.println("next");
        subseqNo = dis.readInt();
        System.out.println("next");
        this.seconds = dis.readInt();
        System.out.println("finished");
    }

    public int getMessageType(){
        return INITIALIZE_INSTRUMENT_REPLY;
    }

    public int getSeconds() {
        return seconds;
    }

    public byte[] cerealize() throws IOException {
        ByteArrayOutputStream bos= new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(bos);
        this.cerealize(dos);
        return bos.toByteArray();
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
        s.writeInt(channelNumber);
        s.writeBoolean(endOfData);

        s.writeInt(subseqNo);
        s.writeInt(seconds);
    }

    public String toString(){
        return "%<msgType=InitializeInstrumentReply%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%channelNumber="
            + this.channelNumber
            + "%endOfData="
            + this.endOfData
            + "%subseqNo="
            + this.subseqNo
            + "%seconds="
            + this.seconds
            + "%>";
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public abstract class Message extends Object {
    public static final int SET_CURRENT_LIMIT_CMD = 10;
    public static final int ACK_REPLY = 11;
    public static final int START_STREAMING_CMD = 12;
    public static final int STOP_STREAMING_CMD = 13;
    public static final int STREAMING_DATA_REPLY = 14;
    public static final int POLL_CMD = 15;
    public static final int POLLED_DATA_REPLY = 16;
    public static final int ERROR_REPLY = 18;
    public static final int ECHO_CMD = 19;
    public static final int ECHO_REPLY = 20;
    public static final int INITIALIZE_INSTRUMENT_CMD = 21;
    public static final int INITIALIZE_INSTRUMENT_REPLY = 22;
    protected int seqNo;    // the sequence number
    protected int len;      // length (in bytes) beyond this

    public abstract int getMessageType(); // one of the above types

    // real classes must implement
    public abstract byte[] cerealize() throws IOException;

    protected void cerealize(DataOutputStream s) throws IOException {
        // CHANGE THIS to lay down the first three parameters,
        //          type, seqNo, and len.
        s.writeInt(getMessageType());
        s.writeInt(seqNo);
        s.writeInt(len);
    }
    public void decerealize(DataInputStream s) throws IOException {
        seqNo = s.readInt();
        len = s.readInt();
    }
    public abstract String toString();
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class PollCmd extends Command {
    public int channelNumber;

    public PollCmd(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public PollCmd(int channelNumber) {
        this.seqNo = getNextCmdSeqNo();
        this.len = 4; // for the channelNumber
        this.channelNumber = channelNumber;
    }

    public int getMessageType(){
        return POLL_CMD;
    }

    public byte[] cerealize() throws IOException {
        ByteArrayOutputStream bos= new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(bos);
        this.cerealize(dos);
        return bos.toByteArray();
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
        s.writeInt(channelNumber);
    }

    public void decerealize(DataInputStream s) throws IOException {
        super.decerealize(s);
        channelNumber = s.readInt();
    }

    public String toString(){
        return "<%msgType=PollCmd%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%channelNumber="
            + this.channelNumber
            + ">%";
    }
}
```



```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class PolledDataReply extends Reply {
    int channelNumber;
    boolean endOfData;
    int subseqNo;
    byte[] data;

    public PolledDataReply(Message m, byte[] data) {
        PollCmd pc = (PollCmd)m;
        this.seqNo = pc.seqNo;
        this.len = 4+1+4+data.length;
        this.channelNumber = pc.channelNumber;
        this.endOfData = true;
        this.subseqNo = 0;
        this.data = data; // no reason to copy it
    }

    public PolledDataReply(DataInputStream dis) throws IOException {
        System.out.println("in ctor(dis)");
        //      this.decerealize(dis);
        //      decerealize(dis);
        //      }

        //      public void decerealize(DataInputStream dis) throws IOException {
        System.out.println("about to superdecereal");
        super.decerealize(dis);
        System.out.println("next");
        channelNumber = dis.readInt();
        System.out.println("next");
        endOfData = dis.readBoolean();
        System.out.println("next");
        subseqNo = dis.readInt();
        System.out.println("about to allocate data for Polled... data");
        this.data = new byte[this.len-4-1-4];
        System.out.println("about to read into it");
        dis.read(data);
        System.out.println("finished reading into data");
    }

    public int getMessageType(){
        return POLLED_DATA_REPLY;
    }

    public byte[] getData() {
        return data;
    }

    public byte[] cerealize() throws IOException {
        ByteArrayOutputStream bos= new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(bos);
        this.cerealize(dos);
        return bos.toByteArray();
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);

        s.writeInt(channelNumber);
        s.writeBoolean(endOfData);
        s.writeInt(subseqNo);
        if (data != null) {
            System.out.println("Cerealizing PolledDataReply, data.length="
                               + data.length);
        } else {
            System.out.println("Cerealizing PolledDataReply, data is null");
        }
        s.write(data, 0, data.length);
    }

    public String toString(){
        char[] better = new char[data.length];
        for (int i=0; i<data.length; i++) {
            better[i] = (char)data[i];
        }
        String theData = new String(better);

        return "%<%msgType=PolledDataReply%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%channelNumber="
            + this.channelNumber
            + "%endOfData="
            + this.endOfData
            + "%subseqNo="
            + this.subseqNo
            + "%data="
            + theData
            + "%>%";
    }
}
```

02/24/02
16:03:26

nmc\common\Reply.java

1

```
package nmc.common;
```

```
public abstract class Reply extends Message {  
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class SetCurrentLimitCmd extends Command {

    public int channelNumber;
    public int currentLimit;

    public SetCurrentLimitCmd(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public SetCurrentLimitCmd(int channelNumber, int currentLimit) {
        this.seqNo = getNextCmdSeqNo();
        this.len = 8;
        this.channelNumber = channelNumber;
        this.currentLimit = currentLimit;
    }

    public int getMessageType() {
        return SET_CURRENT_LIMIT_CMD;
    }

    public byte[] cerealize() throws IOException {
        ByteArrayOutputStream bos= new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(bos);
        this.cerealize(dos);
        return bos.toByteArray();
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
        s.writeInt(channelNumber);
        s.writeInt(currentLimit);
    }

    public void decerealize(DataInputStream s) throws IOException {
        super.decerealize(s);
        channelNumber = s.readInt();
        currentLimit = s.readInt();
    }

    public String toString(){
        return "<%msgType=SetCurrentLimitCmd%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%channelNumber="
            + this.channelNumber
            + "%currentLimit="
            + this.currentLimit
            + "%>%";
    }
}
```